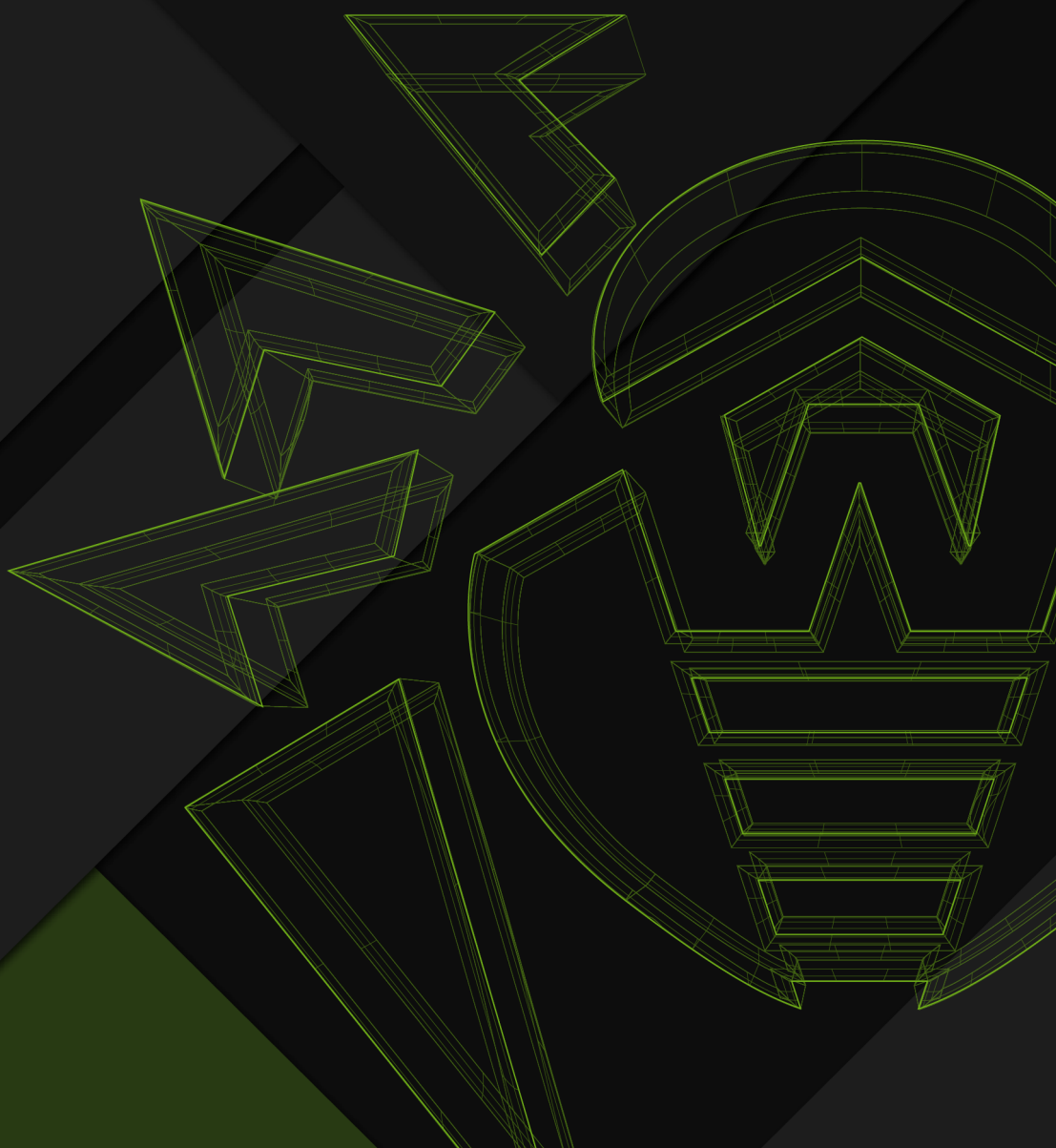




Spear phishing campaigns threaten Russian fuel and energy companies



© **Doctor Web, Ltd., 2020. All rights reserved.**

This document is the property of Doctor Web, Ltd. (hereinafter - Doctor Web). No part of this document may be reproduced, published or transmitted in any form or by any means for any purpose without proper attribution.

Doctor Web develops and distributes Dr.Web information security solutions which provide efficient protection from malicious software and spam.

Doctor Web customers can be found among home users from all over the world and in government enterprises, small companies and nationwide corporations.

Dr.Web antivirus solutions are well known since 1992 for continuing excellence in malware detection and compliance with international information security standards. State certificates and awards received by the Dr.Web solutions, as well as the globally widespread use of our products are the best evidence of exceptional trust to the company products.

Spear phishing campaigns threaten Russian fuel and energy companies
9/24/2020

Doctor Web Head Office
2-12A, 3rd str. Yamskogo polya
Moscow, Russia
125040

Website: www.drweb.com
Phone: +7 (495) 789-45-87

Refer to the official website for regional and international office information.

Table of Contents

| | |
|--|-----------|
| The scouting | 4 |
| The attack | 6 |
| Conclusion | 7 |
| Operating Routine of Discovered Malware Samples | 8 |
| BackDoor.Whitebird.23 | 8 |
| Trojan.DownLoader34.31724 | 14 |
| BackDoor.Siggen2.3244 | 15 |
| BackDoor.Siggen2.3238 | 18 |
| Appendix. Indicators of Compromise | 23 |

The scouting

At the end of April 2020, Doctor Web virus analysts detected a spear phishing campaign where employees of Russian fuel and energy companies were receiving emails with bogus attachments. These emails contained `.docx` files that were distributed under the guise of an updated telephone book and downloaded two images from remote resources.

One of the images was uploaded to a user's PC from `news[.]zannews[.]com`. It is noteworthy that this domain name is similar to the domain of the anti-corruption media center of Kazakhstan—`zannews[.]kz`. On the other hand, this domain instantly recalled another well-known campaign from 2015, known as TOPNEWS, where the ICEFOG backdoor was used, and trojan C&C domains had the "news" substring in their names. Another interesting feature was the use of different request parameters or unique image names in download requests in those cases when emails were sent to different recipients. We believe this was done to collect information and identify a "reliable" recipient who is guaranteed to open an email in subsequent attacks. The SMB protocol was used to download the image from the second server, which could be done to collect NetNTLM hashes from the employees' computers who opened the received document.



In June 2020, the attackers started to use a new domain name—`sports[.]manhajnews[.]com`. Our analysis revealed that `manhajnews[.]com` subdomains have been used in spear phishing campaigns since at least September 2019. Thus, one of the targets of this campaign was a large Russian university.

In addition, another campaign was launched in June. This time, the attached document contained information about industry development. The text in the email clearly indicated the author was not a native Russian speaker. With that, the remote server was changed to `download[.]inklingpaper[.]com`

As an attempt to avoid the anti-virus detection of malicious documents, in July, hackers began using Microsoft Word documents encrypted with a password. At the same time, they have slightly modified the content of the email.



The text was again written in the same style, which caused additional suspicion in the recipient. The remote server for image downloading did not change either.

It is worth noting that in all cases mentioned above, the attackers used mailboxes registered on the mail[.]ru and yandex[.]ru domains.


The first of them—[BackDoor.Siggen2.3238](#)—has not been previously encountered by our team. Any mention of this malware by other anti-virus vendors has also not been found. The second backdoor is a modification of **BackDoor.Whitebird**, which is already known to us from the [incident with the state institution in Kazakhstan](#). Additionally, during our research we located another domain name that was used during this spear phishing campaign—news[.]microotf[.]com. However, we have not found any trojans using this domain as a C&C server.



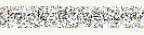
Conclusion

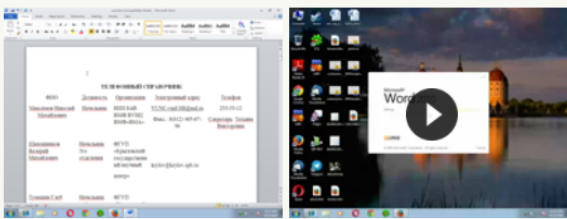
Analysis of documents, malware, and the infrastructure used allows us to say the attack was conducted by one of the Chinese APT groups. Given the functionality of backdoors installed on victims' computers in the event of a successful attack, the infection would allow hackers to at least steal confidential information from compromised computers of the attacked organizations. In addition, a very likely scenario of a further attack is the installation of specialized trojans on local servers of high importance, such as domain controllers, mail servers, Internet gateways, etc. As we saw in the example of the [incident in Kazakhstan](#), these servers are of particular interest to attackers for various reasons.

Doctor Web malware analysts recommend users pay extra attention to all incoming correspondence, including emails received from allegedly known addressees.

A harmful attachment containing a malicious macro was successfully detected by the [Dr.Web vxCube](#) cloud-based analyser.

 **телефонный справочник.doc#**

| | |
|------------------|---|
| Estimated result | Clean  Malware  |
| Detected | Malicious behavior |
| | Threats in files and dumps |
| Size | 187.5 KB |
| Format | DOC |
| SHA1 | a6b7ff870a5105cc0f82fa2d76c4aafbbcad874e |
| User |  |



Operating Routine of Discovered Malware Samples

BackDoor.Whitebird.23

A backdoor written in C++ and designed to operate in both 32-bit and 64-bit Microsoft Windows operating systems. **BackDoor.Whitebird.23** is designed to establish an encrypted connection with the C&C server and for unauthorized control over an infected computer.

Operating routine

Start of operation

The malicious library has two export functions:

```
GooglePlay  
Test
```

At the beginning of its operation, the backdoor decrypts the hardcoded configuration using an algorithm based on the XOR operation with byte 0x99. The configuration:

```
struct st_cfg  
{  
    _DWORD dword0;  
    wchar_t campaign[64];  
    wchar_t cnc_addr[256];  
    _DWORD cnc_port;  
    wchar_t cnc_addr2[100];  
    wchar_t cnc_addr3[100];  
    _BYTE working_hours[1440];  
    wchar_t proxy_domain[50];  
    _DWORD proxy_port;  
    _DWORD proxy_type;  
    _DWORD use_proxy;
```



```
_BYTE proxy_login[50];  
  
_BYTE proxy_password[50];  
  
_BYTE gapa8c[256];  
  
};
```

To ensure the backdoor is always running, it changes the value specified in the `working_hours` field of the configuration. The field contains 1440 bytes, which take the values 0 or 1 and represent every minute of every hour in a day.

The program creates a separate thread for each network interface that listens to that interface and searches for authorization packets on the proxy server from the infected computer. When such a packet is detected, the backdoor adds information about the proxy server to its own list. It also checks for proxy via the `InternetQueryOptionW` WinAPI. The list with proxy information has the following structure:

```
struct st_proxy  
{  
    st_proxy *next;  
    st_proxy *prev;  
    _DWORD proxy_type;  
    _DWORD proxy_addr;  
    unsigned __int16 proxy_port;  
    _DWORD got_proxy;  
    const char login[255];  
    const char password[255];  
    const char char216[255];  
};
```

Proxy server configurations are additionally saved in the `%TEMP file%\<computer_name>_r.xra` file.

The program then checks the current minute and hour and compares them with the data in the `working_hours` field of the configuration. If the value for the corresponding minute of the day is not zero, it establishes a connection to the C&C server.

If the connection to the server is successful, the backdoor collects information about the infected computer and forms the following structure:

```
#pragma pack(push, 1)

struct st_info
{
    wchar_t campaign[64];

    _DWORD dword80;

    WCHAR username[64];

    wchar_t macs[20];

    RTL_OSVERSIONINFOW osversion;

    _BYTE gap130[8];

    WCHAR computer_name[16];

    _BYTE gap24A[96];

    st_drive_info drives[26];

    _BYTE gap2D0[312];

    _DWORD mem_total_phys;

    _DWORD mem_available_phys;

    DWORD cpu_nop;

    DWORD cpu_freq;

    _BYTE cpu_info[128];

    _DWORD x64;

    WCHAR locale[6];
};

#pragma pack(pop)
```

The program sends the collected information to the C&C server and waits for 2 bytes in response. The first of these two bytes the backdoor will send after processing the server command.

Then it sends the byte 0x10, which is a command request from the C&C server. In response, the server must send 16 bytes, where the first `DWORD` serves as the command ID.

The list of commands is shown in the following table:

| Command id | Action |
|------------|--|
| 0x01 | To send a list of files in the folder |
| 0x15 | To delete a file |
| 0x16 | To delete a directory |
| 0x17 | To move a file |
| 0x18 | To open a file |
| 0x19 | To write a file |
| 0x1b | To read a file |
| 0x32 | To send a list of processes |
| 0x33 | To terminate a specified process |
| 0x50 | To send a list of services |
| 0x51 | To send a configuration of the specified service |
| 0x52 | To run a service |
| 0x53 | To stop a service |
| 0x54 | To delete a service |
| 0x64 | To run the command shell with I/O redirecting to pipes or run the command in the command shell |
| 0x65 | To establish an additional connection with the C&C server and run the command shell with I/O redirecting to socket |
| 0x3e7 | To remove itself from the system |

```
v3 = time(0);
```

```
t = (v3 >> 8 >> 16) + ((((((unsigned __int8)v3 << 8) + BYTE1(v3)) << 8) +
BYTE2(v3)) << 8);

for ( i = 0; i < 28; i += 4 )

    *(_DWORD *)&clientrnd[i] = t + *(_DWORD *)&cnc_addr[i / 4];

for ( j = 0; j < 28; ++j )

    clientrnd[j] ^= 7 * (_BYTE)j;
```

The resulting packet is sent to the C&C server. The response (Server Hello packet) is checked for compliance with the following parameters:

- If the TLS Protocol version corresponds to version 1.0;
- The match between the timestamp (the first 4 bytes of the Random Data field) specified by the client and the timestamp specified by the server;
- The match between the first 4 bytes after the timestamp in the Random Data field specified by the client and these 4 bytes specified by the server.

In the case of these matches, the backdoor prepares the Client Key Exchange packet. To do this, it modifies Public Key in the Client Key Exchange packet, as well as the Encryption IV and Encryption Data in the Encrypted Handshake Message packet.

```
z = 17;

x = 17 * t;

do

{

    client_key_exchange[z++] ^= BYTE1(x);

    x += t;

}

while ( z < 272 );

p = &next_after_client_key_exchange;

c = 7;

do

{

    *p ^= BYTE1(c);

    c += t;
```

```
--p;  
  
}  
  
while ( (int)p > (int)&client_key_exchange[278] );
```

The backdoor then receives the packet from the C&C server, checks that the TLS Protocol version corresponds to version 1.0, and then receives another 54 bytes (the packet body). This completes the connection establishment.

All received and sent packets are transmitted over the TLS 1.0 protocol, and the encryption of these packets is determined by the following algorithm:

```
for ( i = 0; i < size; ++i )  
  
    buffer[i] ^= 7 * (_BYTE)i;
```

If the client or server failed to accept a packet in one attempt, it sends back a packet, which is formed as follows:

```
t = time(0);  
  
...  
  
for ( j = 0; j < 10; ++j )  
  
    Src[j] = (unsigned __int16)(0x71 * (t + 35)) >> 8;  
  
tls_send_packet(socket, Src, 0xAu);
```

Trojan.DownLoader34.31724

Trojan.DownLoader34.31724 is a shellcode loader written in C++. It works both in 32-bit and 64-bit Microsoft Windows operating systems.

Operating routine

The C&C server address is hardcoded in trojan's body and is plain text: `https://newsfor[.]newss[.]nl`.

Using the `CoCreateGuid` function, the program generates a GUID (Globally Unique Identifier), which is then used as the identifier of the infected device.

It then sends a GET request to the C&C server with the `User-Agent: "WinHTTP Example/1.0"` HTTP header to `https://newsfor[.]newss[.]nl/<uuid>&0&4`, where:

- `<uuid>`—previously created `uuid`,
- 0—the sequence number of the request,
- 4—a number, hardcoded in the trojan's body.

In response to this GET request, the trojan waits for the shellcode delivery, and then sends another GET request. If the first `DWORD` in the response is equal to 1, it then executes the previously received shellcode.

The result of the shellcode execution is sent by a POST request to the same URL that is used for GET requests.

BackDoor.Siggen2.3244

A trojan dropper designed to spread and install other malicious applications onto the targeted computers. The malware is written in C and supports 32-bit and 64-bit Microsoft Windows operating systems. The analyzed sample was used to spread the **BackDoor.Whitebird.23** trojan.

Operating routine

The sample in question was distributed inside the SFX RAR archive containing the following files:

- `migwiz.exe`—the main dropper
- `migwiz3.DAT`—an encrypted shellcode with the payload for 32-bit Microsoft Windows operating systems
- `migwiz6.DAT`—an encrypted shellcode with the payload for 64-bit Microsoft Windows operating systems

Main dropper `migwiz.exe`

Upon launching, it enables the following system privileges for its own process:

- `SeDebugPrivilege`
- `SeAssignPrimaryTokenPrivilege`
- `SeBackupPrivilege`
- `SeRestorePrivilege`

Next, depending on the bitness of the operating system, it reads the `migwiz3.DAT` or `migwiz6.DAT` file located in the same directory from where the trojan was launched. If it has administrator rights, it saves the contents of the corresponding file into the `[HKLM\Software\Microsoft\MigWlz\Options]` registry key with the `Setup` value. If it

doesn't have appropriate rights, it saves the file contents into the [HKCU\\Software\\Microsoft\\MigWlz\\Options] registry key with the Setup value.

These keys store the payload that the trojan will extract and launch later (see the "The payload" section below).

If the dropper was launched with administrator rights, it sets an autorun option for the IKEEXT service. It then checks if the %WINDIR%\\System32\\wlbctrl.dll file exists. If the file exists, the trojan fills it with random data, deletes the file and replaces it with the new one, which is encrypted and packed through the RtlCompressBuffer and stored in its body. To decrypt the file, it uses the following script:

```
def decrypt(data):  
  
    a = b = c = d = struct.unpack('<I', data[:4])[0]  
  
    s = bytearray()  
  
    for i in range(len(data)):  
        a = (a + (a >> 3) - 0x3C6312B6) & 0xffffffff  
        b = (b + (b >> 5) - 0x62B652F6) & 0xffffffff  
        c = (c + 0x6251F3E2 - (c << 9)) & 0xffffffff  
        d = (d + (0xCD56823E - (d << 7))) & 0xffffffff  
  
        s.append(data[i] ^ ((a + b + c + d) & 0xff))  
  
    return s
```

With that, the trojan writes a large block of the random data into the end of the extracted file. This block starts with the 0xcc90 signature and has the size of 0x8FF1B6 bytes for the 64-bit library and the size of 0x8FF437 bytes for the 32-bit library.

The trojan sets the same date of creation, access and modification for the extracted library as the %WINDIR%\\System32\\rundll32.exe system file has. After that, it launches the IKEEXT service, which loads the malicious library into the memory.

If the trojan was launched without administrator rights, it checks if the %ProgramData%\migwiz directory exists. If it exists, the trojan fills every file in it with the random data and deletes these files. Next, it deletes the directory itself.

After that, the trojan recreates this directory and places the migwiz.exe and migwiz.dll files, which it extracts from its body, into this catalogue. Herewith, the library migwiz.dll represents the same library that is saved into the %WINDIR%\System32\wlsctrl.dll directory if the trojan has the appropriate administrative rights.

The trojan then adds the migwiz.exe file to the autorun list, creating the [HKCU\Software\Microsoft\Windows\CurrentVersion\run] registry key with the migwiz parameter.

For the migwiz.exe and migwiz.dll files, as well as for the directory where they are located, the trojan sets the same date of creation, access and modification as the rundll32.exe system file has. It then launches the migwiz.exe file.

migwiz.exe loader

Searches for the dll libraries that are located in the same directory with it and loads them into the memory.

migwiz.dll loader

Uses the mutex to ensure only one copy of the trojan is running at the same time. If the trojan is launched as a service, the mutex has the Global/Config name. In other cases, the mutex has the LocalConfig name.

Depending on how the trojan is launched, the loader reads the payload from the [HKLM\Software\Microsoft\MigWlz\Options] 'Setup' or [HKCU\Software\Microsoft\MigWlz\Options] 'Setup' registry keys. If the reading was successful, the payload is decrypted with the XOR operation and the 0x90 byte.

Next, if the trojan was launched as a service, it injects the payload into the running explorer.exe process. Otherwise, it injects the payload into its own process.

The payload

The payload is stored in the Windows registry, where it is placed at the initial stage of infection after it was read from the migwiz3.DAT and migwiz6.DAT files. It contains the shellcode that obtains the addresses of the minimum number of functions required by the trojan and loads it from its own body as an MZPE file. This file represents a dll library detected by Dr.Web as **BackDoor.Whitebird.23**.

BackDoor.Siggen2.3238

A backdoor trojan for 32-bit Microsoft Windows operating systems. It is written in C++. Its main functionality is to obtain unauthorized access to infected computers and perform malicious actions on behalf of attackers.

Operating routine

Upon launching, **BackDoor.Siggen2.3238** initiates a series of preliminary checkups. First, it receives the addresses of the following exported functions:

- CreateDirectoryExW
- NtQueryDirectoryFile
- NtDeleteFile
- NtWriteFile
- NtReadFile
- NtCreateFile
- NtSetInformationFile

Next, it verifies the pointer to each function using the algorithm as follows:

```
1 int __cdecl magic(unsigned __int8 *a1)
2 {
3     if ( *a1 == 0xE9
4         || *a1 == 0x90 && a1[1] == 0xE9
5         || *a1 == 0x8B && a1[1] == 0xFF && a1[2] == 0xE9
6         || *a1 == 0x68 && a1[5] == 0xC3
7         || *a1 == 0x90 && a1[1] == 0x68 && a1[6] == 0xC3
8         || *a1 == 0xFF && a1[1] == 0x25
9         || *a1 == 0x8B && a1[1] == 0xFF && a1[2] == 0xFF && a1[3] == 0x25
10        || *a1 == 0xB8 && a1[5] == 0xFF && a1[6] == 0xE0
11        || *a1 == 0xB8 && a1[5] == 0x50 && a1[6] == 0xC3
12        || *a1 == 0xA1 && a1[5] == 0xFF && a1[6] == 0xE0
13        || *a1 == 0xA1 && a1[5] == 0x50 && a1[6] == 0xC3
14        || *a1 == 0x90 && a1[1] == 0x90 && a1[3] == 0xE9
15        || a1[5] == 0xFF && a1[6] == 0x25 )
16     {
17         sub_42DFF0();
18     }
19     return 0;
20 }
```

If the data at the pointer matches the one to be checked, the trojan allocates a memory region in the size of 1 byte, fills 10 bytes in this region with zeroes and tries to free it:

```
1 void sub_42DFF0()
2 {
3     _DWORD *v0; // [esp+4h] [ebp-8h]
4
5     v0 = operator new[](1u);
6     *v0 = 0;
7     v0[1] = 0;
8     *((_WORD *)v0 + 4) = 0;
9     j_j__free(v0);
10 }
```

This operation is performed for each exported function. By doing so, the backdoor most likely checks for the active hooks in the functions and attempts to terminate the active process.

BackDoor.Siggen2.3238 then checks if the `false` and `true` strings hardcoded in its code are matching. If they match, the trojan checks for the VMWare and VirtualBox components, namely VMWare Guest Additions, Virtual Machine Additions and Virtualbox Guest Additions, are present in the system. After this verification is complete, its results are reset, and the trojan enters an endless loop where it doesn't perform any other action.

```
21 p_false = "false";
22 p_true = aTrue;
23 while ( 1 )
24 {
25     c = *p_true;
26     v4 = c < (unsigned int)*p_false;
27     if ( c != *p_false )
28         break;
29     if ( !c )
30         goto LABEL_6;
31     d = p_true[1];
32     v4 = d < (unsigned int)p_false[1];
33     if ( d != p_false[1] )
34         break;
35     p_true += 2;
36     p_false += 2;
37     if ( !d )
38     {
39 LABEL_6:
40         v9 = 0;
41         goto LABEL_8;
42     }
43 }
44 v9 = v4 ? -1 : 1;
45 LABEL_8:
46 v8[1] = v9;
47 if ( !v9 )
48 {
49     check_vmwarehostopen_exe();
50     check_svc_1_driver_vmsrv();
51     check_vb_guest_additions();
52     v12 = 0;
53     is_vm = 0;
54     infinite_loop();
55 }
```

The main functionality

BackDoor.Siggen2.3238 can connect to the C&C server using both HTTP and HTTPS protocols. The analyzed trojan sample uses the HTTPS protocol. When sending the requests to the server, the following User-Agent is used:

Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0; SE)

With that, all the requests are sent with the set of the parameters as follows:

%s;type=%s;length=%s;realdata=%send

where each %s string is correspondingly replaced by the strings shown below:

- the infected computer ID
- the type of the request to be sent
- the length of the data in the `realdata` field
- the data

After the initial verification is complete, **BackDoor.Siggen2.3238** generates its own ID using the following function:

```
1 DWORD *gen_bot_id()
2 {
3     unsigned int v0; // eax
4     _BYTE *botid; // [esp+Ch] [ebp-4Ch]
5     int i; // [esp+10h] [ebp-48h]
6     char a[64]; // [esp+14h] [ebp-44h] BYREF
7
8     qmemcpy(a, "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz", 62);
9     botid = operator new[](30u);
10    v0 = j_time64(0);
11    srand(v0);
12    *(_DWORD *)botid = 0;
13    *((_DWORD *)botid + 1) = 0;
14    *((_DWORD *)botid + 2) = 0;
15    *((_DWORD *)botid + 3) = 0;
16    *((_DWORD *)botid + 4) = 0;
17    *((_DWORD *)botid + 5) = 0;
18    *((_DWORD *)botid + 6) = 0;
19    *((_WORD *)botid + 14) = 0;
20    for ( i = 0; i < 24; ++i )
21        botid[i] = a[rand() % 62];
22    return botid;
23 }
```

Next, it collects the information about the infected system and forms the string shown below:

```
lan=%s;cmpname=%s;username=%s;version=%s;
```

where `lan` is the IP address of the infected computer, `cmpname` is the name of the computer, `username` is the user name, and `version` is a `0.0.4.03` string.

This information, paired with the `sysinfo` ID, is sent to the C&C server located at the <https://31.214.157.14/log.txt>. If **BackDoor.Siggen2.3238** receives the `HEART` signal in response, the connection is considered successful, and the trojan proceeds to the main cycle of communication with the server.

To receive new commands, the backdoor sends the packet with the `HEART` command ID with the `heart` data. The server response that follows is parsed with the regular expression shown below:

```
type=([^\&]+);first=([^\&]+);second=([^\&]+);third=([^\&]+);
```

The `type` string in this response characterizes which command needs to be executed, while other strings contain the parameters for these commands.

BackDoor.Siggen2.3238 can receive the following commands:

| Command | Description |
|---------|---|
| HEART | Heartbeat (a beacon signal that keeps the C&C server and the backdoor connected). |

| | |
|------------------|---|
| OK | A server confirmation indicating the data sent by the trojan has been received successfully. |
| CMDINFO | To launch the cmd.exe with the input-output redirection into the pipes, through which the data is sent to the server and back. |
| PROCESSINFO | To collect information about the running processes. The information about each process is represented as proName=%1%;PID=%2%;proPath=%3%;Tport=%4%;Uport=%5%;descrip=%6%;. |
| PROCESSTERMINATE | To kill the process with the specified PID. |
| LISTDRIVE | To collect information about disks. The information about each disk is represented as diskName=%s;driveType=%s;. |
| LISTFILE | To collect the listing of the specified directory. The information about each file or catalogue in it is represented as fileName=%1%;path=%2%;fileType=%3%;fileSize=%4%;access=%5%;create=%6%;. |
| DELETEFILE | To delete the specified file. |
| DOWNLOAD | To upload the specified file onto the server. |
| UPLOAD | To download the specified file from the server. |
| RUN | To launch the specified file. |

Appendix. Indicators of Compromise

SHA1 hashes

BAT.Starter.318

07bdaa2ef4556d9c14753c53c7fc239e9e669637: configstest.bat

Trojan.DownLoader34.31724

091866cac1bef518dbb6d114b3636fbad144b49a: rdplib64.exe

8e2c253615e3e49e81e43a28d5b0d2a7fc54ac2b

bb373b8a81deaccc4f69cac3bde0d6174b261f37

bbb29d96809bcd4c0e75df8f08f3e9dbc817f584: rdplib.exe

BackDoor.Siggen2.3238

3884263dfe67a3da0079fe40d6186950b853145c

BackDoor.Siggen2.3244

c36aabe2828b84a1221a8855b984187b89c24b44: dlhost.exe

632f6737f5308b49cc198fea88338a3403732274: migwiz.exe

a2cab5d0c2a7eb93e24c32c407059464dc66ab97: migwiz.dll

7b9e9c67f42671d33c9e7d4d7a36231f1de49bb7: migwiz.dll

02676f335b800ff1c42a1f4fe2344ac381d914f1: migwiz.exe

ebb1c0ad2ad2bcdecf5182be7bd3ea5b18cc2126: migwiz.exe

BackDoor.Whitebird.23

2510e873e79cfb61533e9b5a124ddbec130c653c: migwiz3.DAT

d6e84ad926cc1d5a3d300a98f492380a31b2427b: migwiz6.DAT

Domains

newsinfo[.]newss[.]nl

newsfor[.]newss[.]nl

news[.]newss[.]nl

webnews[.]newss[.]nl

nissen[.]newss[.]nl

john[.]newss[.]nl

news[.]microotf[.]com

news[.]zannews[.]com

download[.]inklingpaper[.]com

sports[.]manhajnews[.]com

gova[.]manhajnews[.]com

duck[.]manhajnews[.]com

IPs

185.158.249[.]120

109.230.199[.]173

109.230.199[.]138

109.230.199[.]124

109.230.199[.]148

176.10.118[.]154

176.10.125[.]159

31.214.157[.]114

31.214.157[.]126

122.10.82[.]165